CollegeBoard
Advanced Placement Program

Student Performance Q&A:

2007 AP® Computer Science A Free-Response Questions

The following comments on the 2007 free-response questions for AP® Computer Science A were written by the Chief Reader, David Reed of Creighton University in Omaha, Nebraska. They give an overview of each free-response question and of how students performed on the question, including typical student errors. General comments regarding the skills and content that students frequently have the most problems with are included. Some suggestions for improving student performance in these areas are also provided. Teachers are encouraged to attend a College Board workshop to learn strategies for improving student performance in specific areas.

Question 1

What was the intent of this question?

This question focused on algorithm design and implementation, as well as array manipulation. A property of integers was described, that of being a self-divisor (when every digit of the number evenly divides the number). In part (a) students were required to implement the `isSelfDivisor` method for determining whether a given number is a self-divisor. This could be accomplished numerous ways, e.g., by repeatedly extracting the rightmost digit (using the remainder operator) and then dividing by 10, or by converting the number to a string and then extracting digits as characters. In part (b) students were required to implement a method for finding and collecting self-divisors in an array. The `firstNumSelfDivisors` method had two parameters, the starting number in a range and the number of desired self-divisors. The method was intended to find self-divisors from the range, place them in an array, and return that array. This involved looping through numbers, starting at the start number, and calling the `isSelfDivisor` method from part (a) to identify self-divisors.

How well did students perform on this question?

This question was comparable to A1 questions in previous years in terms of its difficulty. It did require more of the student in terms of algorithm design, however, as there were numerous ways that part (a) could be implemented. As was the case with all the A exam questions, there were many zeros and blanks (almost 25 percent), and very few 9s due to the amount of detail in the question. Otherwise, the scores were fairly evenly distributed. Overall, the question had the second highest mean on the exam: 3.86 out of 9. With zeros and blanks removed, the mean was a strong 5.13, suggesting that students who made a serious effort were able to perform reasonably well.

What were common student errors or omissions?

The less-constrained format of this question clearly gave some students problems. In part (a) a variety of approaches were taken in attempting to extract a digit. While most students did use the remainder operator, a sizeable number converted the number into a string and then extracted digits as characters, although these attempts usually contained errors (such as failing to convert the digit back into a number before dividing). A surprising number of students made their solutions unnecessarily complex, extracting digits, storing those digits in a separate list, and then traversing the list. Students performed better on part (b), although there was still variety in the approaches taken. Some students chose to structure their code with a single loop, while others used nested loops to step through the range and find the next self-divisor. Common errors included off-by-one errors in the loop (e.g., adding one too many self-divisors to the list), hard-coding the number of repetitions (e.g., using the constant 3 from the sample in the text), and failing to return the constructed list.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

Students need to be comfortable in taking a problem description, designing a solution, and implementing that solution. Often, more than one solution is possible, so being able to identify a straightforward approach and follow through on the details are important skills being tested. For this question, the two most common approaches in part (a) were not equivalent in terms of coding difficulty. The string conversion approach led to more complexity and more errors, so a student who was able to identify the more straightforward approach to the problem was more likely to avoid errors.

Question 2

What was the intent of this question?

This question was based on the Marine Biology Simulation (MBS) Case Study and focused on abstraction and inheritance. Students needed to show their understanding of the case study and its interacting classes by writing member functions for a new `PounceFish` class. In part (a) students were required to implement the private `findFish` method, which searched ahead to find and return the nearest neighboring fish, if there was one within range. In part (b) the students were required to override the Fish `act` method to produce the desired behavior. This involved calling the `findFish` method from part (a) to determine if a fish was within range, and if so to pounce on that fish (i.e., remove that fish from the environment and move to its old location). If no fish existed within range, it behaved as a normal fish by calling the `super.act()` method.

How well did students perform on this question?

This question was similar to MBS questions from previous years, although perhaps more algorithmically complex due to the need to search ahead in the environment. The distribution of scores was heavily skewed to the low end, with numerous scores in the 0–3 range. As is common on case study questions, there were a large number of zeros and blanks (26 percent), suggesting that some teachers may not be

sufficiently emphasizing the case study in exam preparation. The question had the lowest mean on the exam, only 3.19 out of 9. Even ignoring zeros and blanks, the mean was only 4.33.

What were common student errors or omissions?

The most common errors in part (a) were related to searching ahead in the environment to find a fish. Malformed loops were common, as were attempts to hard-code searching in the four possible directions. Some students tried to avoid the `getNeighbor` method altogether by calculating row and column offsets from the current location, usually with errors resulting. The return in part (a) was also commonly missed, with many students returning the location of the found fish instead of the fish itself. Student performance was better in part (b), which was algorithmically much simpler. While most students recognized the need to call the `findFish` method from part (a), it was often used incorrectly (e.g., treating the return type as a Boolean).

Based on your experience at the AP Reading, what message would you like to send to teachers that could improve the performance of their students on the exam?

Teachers who are not covering the case study or who are relegating it to the very end of the year need to recognize its importance. Starting in 2008, familiarity with the new GridWorld case study will be expected of all students. There will be a free-response question and several multiple-choice questions based on the case study every year. These questions will depend upon students being familiar with and comfortable using classes from the case study.

Question 3

What was the intent of this question?

This question focused on abstraction, ArrayList traversal, and the application of basic algorithms. Students were provided with the framework of a `StudentAnswerSheet` class, which represents a sequence of answers to a multiple-choice test. In part (a) students were required to implement the `getScore` method, which takes an answer key (an ArrayList of Strings) and determines the score for the given answer sheet. This involved traversing both ArrayLists (the answer sheet and the key), comparing the strings at corresponding indices, and assigning points based on whether they matched. In part (b) the framework for a client class was provided, which stores an ArrayList of answer sheets as a field. Students were required to implement the `highestScoringStudent` method, which finds and returns the name of a student from the ArrayList with highest score. This involved traversing the ArrayList, calling the `getScore` method from part (a) on each answer sheet, identifying the sheet with highest score, accessing the name associated with that sheet, and returning that name.

How well did students perform on this question?

This question was comparable to abstraction and class-use questions in past years. Student performance was strong, with a large number of scores in the 7–9 range. Relative to the other questions, there were few zeros and blanks, suggesting that students at all levels of mastery found some parts of the question that

they could complete. This question had the highest mean on the exam: 5.04 out of 9 (or 6.1 out of 9 if zeros and blanks are ignored).

What were common student errors or omissions?

In part (a) the most common errors tended to be minor (e.g., failing to initialize the score variable, incorrectly comparing string values using `==`, and accessing the ArrayList elements using []). A significant number of students did not handle the "?" case correctly, failing to recognize the blank response and instead treating it as a wrong answer. In part (b) common errors included calling `getScore` with incorrect parameters, comparing only adjacent elements when traversing the ArrayList, and returning the entire sheet instead of just the student's name. A number of students incorrectly assumed that the minimum possible score was 0, and their code failed if all students had negative scores.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

Questions such as this one, where students are given class frameworks and asked to implement specific methods, will continue to be on future exams. Students need to be comfortable with using the methods of a class, even if they do not know the underlying implementation details. In fact, being able to abstract away the details and focus on the behavior of provided methods is an important problem-solving skill that is being tested. As this question demonstrates, common algorithmic tasks, such as finding a minimum value from a list, may be tested in a variety of contexts.

Question 4

What was the intent of this question?

This question focused on abstraction, class design, and inheritance. Students were provided with an abstract framework for representing different types of games, including a `GameState` interface for capturing the state of a particular game and a `Player` class for representing a game player. In part (a) students were required to extend `Player` by designing and implementing a `RandomPlayer` class that always selects its move at random. This involved knowing the syntax of inheritance and also recognizing which methods needed to be overridden. Overriding the `getNextMove` method required calling the `getCurrentMoves` method defined by the `GameState` interface, randomly selecting a move (if one exists), and returning that move. In part (b) students were required to implement the `play` method of a `GameDriver` class, which calls the appropriate `GameState` and `Player` methods to alternate player moves until the game is over.

How well did students perform on this question?

This question was different from previous years in the amount of abstraction involved. Students had to be able to use the `GameState` interface without any specific instantiation being involved. Likewise, the `Player` and `RandomPlayer` classes were defined independent of any particular game. This amount of abstraction was no doubt difficult for some students. The distribution of scores was skewed to the low

end, with a large number scores in the 0–2 range. This question had the second lowest mean on the exam: 3.68 out of 9. Ignoring the large number of zeros and blanks, however, the mean was 4.66, suggesting that more capable students did reasonably well on the question.

What were common student errors or omissions?

The abstract nature of this question gave many students problems, and the fact that it was the last question probably contributed to some blank or hurried solutions. Many students were confused about which methods were internal to a class and which had to be applied to external objects. For example, the object `state` was commonly omitted when calling methods such as `getCurrentMoves` and `getCurrentPlayer`. Likewise, the `Player` methods `getNextMove` and `getName` were commonly called with missing or incorrect objects. In part (a) many student errors occurred when attempting to select a random move. While a variety of approaches were attempted (using `Math.random`, `Random`, and even `RandNumGenerator` from the case study), errors in calling the methods were frequent. In part (b) the fairly detailed pseudocode in the problem description helped students to structure their code, but errors in printing the initial state, accessing the player's name to print, and calling the `makeMove` method were common.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

This question was the ultimate test of a student's ability to use an interface and class framework in an abstract context. Students who were able to focus on the available methods and ignore irrelevant information (e.g., the particular game application, the player strategies), and who were able to systematically follow pseudocode, performed well. This question involved designing and implementing a derived class based on a description of its desired behavior, skills that will continue to be tested on future exams. On a technical note: teachers should realize that the `Random` class has been removed from the APCS Java subset for 2008. Students should be comfortable using `Math.random` to generate random values as needed on future exam questions.